Version ID -- ECSE-155

Author(s): John Farley

Description of Problem

The Version field has traditionally been a free text format field that gave data providers flexibility in defining a Version that reflected their data product and so we see a wide variety in the information stored in the Version field. Unfortunately, this makes it difficult to use the Version field for its original purpose which was disambiguating between different collections of the same data. ECSE-155 attempts to standardize the format of the Version field in a way that clarifies the field contents and enables collection disambiguation.

JIRA Linkage

ECSE-155 - JIRA project doesn't exist or you don't have permission to view it.

Background:

Origin of Version -- In the early days of EOSDIS, the priority was on defining systems to support large flagship missions such as Aqua and Terra. Since the data produced by these flagship missions was expected to be collected and supported for decades, it was recognized that the data products would be periodically reprocessed using new algorithms that incorporated the latest scientific understanding. The resultant data products could be significantly different from the previously created data products. To distinguish between the data produced by different processing algorithms, the ECS data model incorporated a Version that was meant to represent a specific version of the algorithms that were used to produce a new "version" of a Collection.

Growth and diversification of EOSDIS -- As time progressed, EOSDIS collection holdings expanded to include a variety of data products from small missions, aircraft data, ground campaigns, and international partners. Many of these new products were not funded and produced under the EOSDIS banner and were not intended for the same type of applications as the large flagship missions. At the same time, the ECHO system became the prime interface for many of the new products and EOSDIS. Recognizing that the EOSDIS definition of Version as the version of reprocessing algorithms might not fit all cases, the ECHO metadata standards incorporated Version as a free text field. This gave data providers flexibility in defining a Version that reflected their data product and so we see a wide variety in the information stored in the Version field. Unfortunately, this makes it difficult to use the Version field for its original purpose which was disambiguating between different collections of the same data. Even a simple operation ,such as choosing the most recent version of a product, becomes difficult when the data provider has chosen free text as a version id.

Analysis:

What are the uses for Version?

In the context of UMM, CMR, & EOSDIS, Version has 2 primary uses:

1. To disambiguate between different collections of the same data, such as distinguishing between MODIS Version 5 and Version 6 products. In the original role of supporting large flagship missions, EOSDIS Version's served as a useful tool for identifying reprocessed versions of a data product.
2. To provide an ordering of the different collections of the same data. It is especially useful to know which Version of a data Collection is the latest version of the Collection.

What is the current definition of a Version?

ECHO/CMR adopted the legacy method of using free text storage type for Version. The free text field put the responsibility of deciding what should be used for the Version on the data providers. The existing Versions in CMR/ECHO provide us with a large number of examples of what data producers would like to use for Version. An analysis of the existing Versions shows that there are several categories of Versions including:

1. The original EOSDIS definition of Version, where a single integer represents the sequential version number of the algorithms used to produce the data. An example is the recent reprocessing campaign for MODIS data which used Version 6 of the MODIS algorithm package and the resulting Version=6 for all of the reprocessed MODIS collections.
2. An extended definition of the EOSDIS definition which includes a major and minor version of the collection, e.g. Version=6.2
3. A date entry. Typically these dates represent either a campaign period (e.g., Summer 2012), a data collection date (e.g., June 2014) or a processing date (e.g., 5/10/2011).
4. A character string that represents useful version information that is specific to that data product. Examples include: "MPI ECHAM3 (T42 L19) 1994" or "CCSR/NIES AGCM 5.4 02 (T21 L20) 1995"
5. There are a small number of Version strings that are not categorizable, e.g., TurboWin+ or PRISM3D

Is Version useful for disambiguating between different collections of the same data?

Today, because of the lack of a clean definition for Version across the full spectrum of EOSDIS data, Version has become difficult to use in disambiguating data collections. In the future, as CMR begins to support full data access through a variety of services, the current use of Version will no longer answer the primary questions required for disambiguation. For example: if a service offers a subsampled version of MOD08_M3.006 is the subsampled data still of Version 006? If not, what then? The problem is that Version has served as a proxy for the full provenance for the product. A full provenance would disambiguate between the original MOD08_M3.006 and the subsampled version of the data. While Version worked for carefully controlled flagship missions that were rarely reprocessed, it will become less and less useful in the era of automated services that add value to the original data. At that time, either a full provenance will need to be used for disambiguation or a different provenance proxy will need to be developed.

Recommendations:

1) Provide a new definition of Version for UMM-C.

The Version should be strongly typed to enforce the definition and eliminate the problems of unformatted fields. The Version should support the requirement of disambiguating between different collections of the same data. The Version should also support the ability to place collections of the same data into an ordered list, in particular it is important to be able to identify the last processed version of the collection.

The recommended definition for Version is a strongly typed field that includes a major and minor version number. The UMM field will be typed to enforce only numeric major and minor version number in the form 6.0, or 7.2. Minor version numbers will default to 0 if not provided. For example: for MOD08_M3.006 product the version number (i.e., 006) will be mapped to the new UMM-C fields as:

Example MOD08_M3.006 Version:

```
<Version>
    <ProcessingVersion>
        <MajorVersion>6</MajorVersion>
        <MinorVersion>0</MinorVersion>
    </ProcessingVersion>
</Version>
```

The following table demonstrates the mapping of some existing "Version" fields into the new definition of Version for UMM-C. Note that for existing versions that use alphabetic indicators, e.g., 1a, we map the alphabetic character to the index of the character within the alphabet. That is, a->1, b->2, etc. See section 4) below for further guidance on choosing good Version fields.

Current Version definition	New UMM-C Version definition
1.000	/Version/ProcessingVersion/MajorVersion = 1 /Version/ProcessingVersion/MinorVersion = 0
0001	/Version/ProcessingVersion/MajorVersion = 1 /Version/ProcessingVersion/MinorVersion= 0
1a	/Version/ProcessingVersion/MajorVersion = 1 /Version/ProcessingVersion/MinorVersion= 1

0.0.1.b	/Version/ProcessingVersion/MajorVersion = 1 /Version/ProcessingVersion/MinorVersion= 2
2_beta	/Version/ProcessingVersion/MajorVersion = 2 /Version/ProcessingVersion/MinorVersion= 0
alpha1	/Version/ProcessingVersion/MajorVersion = 1 /Version/ProcessingVersion/MinorVersion= 0

2) Provide a new field, ProducerVersionID,

As noted in the analysis section, data producers have placed a large variety of information in the Version field. While this information isn't useful for disambiguating collections, or ordering collections, that information is still valuable. We recommend the creation of a ProducerVersionID field for collections that can be used by producers to store free text strings that give additional information about the version of the data. Examples include: "MPI ECHAM3 (T42 L19) 1994" or "CCSR/NIES AGCM 5.4 02 (T21 L20) 1995". All such fields that are currently maintained in the Version field can be directly migrated to the ProducerVersionID. It will be added as part of the DataIdentification section of UMM-C. (Note: Original field name was localVersionID, but at Metadata Curation Summit 1, it was requested that we change the name to ProducerVersionID as the producers are the ones normally driving the strange naming.

3) Provide additional guidance for providers in creating meaningful Versions.

A new major version number should represent a version of the algorithms that creates a new version of the data containing additional scientific insights. A minor version number should represent an incremental improvement in the algorithms that improve the quality of the data (e.g., new calibration tables) without altering the scientific models of the major version.

The sections below detail the proposed changes based on the analysis done by the ECSE team:

Changes to UMM-C fields

1) The current Version field, /DataIdentification/Version will be changed from a free text field to a typed field with major and minor components. The resultant Version will look like:

/DataIdentification/Version/ProcessingVersion/MajorVersion

/DataIdentification/Version/ProcessingVersion/MinorVersion

2) A new field, ProducerVersionID, will be added to the DataIdentification section of UMM-C. The ProducerVersionID will be a free text field and will not be part of the unique identifier for a collection. The field path will be:

/DataIdentification/ProducerVersionID

Changes to UMM-Common fields

No changes are proposed for UMM-Cmn fields.

Mappings to DIF, ECHO, ISO

While the mappings from DIF/ECHO/ISO to /DataIdentification/Version haven't changed, a data translation activity will be required. The existing Version information will be translated as follows:

- For existing Version information that fits the Major.Minor version format, the metadata will be directly translated. This includes existing Version information that only contains a major number. In that case the translation will include setting the Minor field to zero.
- For existing Version information that does not fit the Major.Minor version format, the translation process will set the new Version field to 1.0 and the contents of the existing Version information will be placed in the /DataIdentification/ProducerVersionID field.

Interoperability Considerations

N/A

Changes to CMR

The translation of the legacy formats into UMM will to take into account the translation process described above.

What should the MMT forms look like

The structure of the forms shouldn't change, but the entry form for Version should require entry of a Major.Minor format. Also, an entry field for LocalVersionId needs to be added to the DataIdentification forms.

How should the values in these fields be presented to the user on the EDSC

Version should be presented as Major.MInor, such as 3.0, or 6.4